Positive first-order logic on words and graphs

Denis Kuperberg

CNRS, LIP, ENS Lyon, Plume Team

Birmingham Theory Seminar
24 November 2023
First-Order Logic (FO)

**Signature**: Predicate symbols \((P_1, \ldots, P_n)\) with arities \(k_1, \ldots, k_n\).

**Syntax** of FO:

\[
\phi, \psi := P_i(x_1, \ldots, x_{k_i}) \mid \phi \lor \psi \mid \phi \land \psi \mid \neg \phi \mid \exists x. \phi \mid \forall x. \phi
\]

**Semantics** of \(\phi\):

Structure \((X, R_1, \ldots, R_n)\) is accepted or rejected.

**Example**: Directed graphs: one binary predicate \(E\).

**Graph class**: Cliques

No node points to everyone

**Formula** \(\phi = \forall x. \forall y. E(x, y)\)

**Formula** \(\psi = \neg \exists x. \forall y. E(x, y)\)

**Example graph**

Model of \(\phi\)

Model of \(\psi\)
First-Order Logic (FO)

**Signature**: Predicate symbols \((P_1, \ldots, P_n)\) with arities \(k_1, \ldots, k_n\).

**Syntax** of FO:

\[ \varphi, \psi ::= P_i(x_1, \ldots, x_{k_i}) \mid \varphi \lor \psi \mid \varphi \land \psi \mid \neg \varphi \mid \exists x. \varphi \mid \forall x. \varphi \]

**Semantics** of \(\varphi\):
Structure \((X, R_1, \ldots, R_n)\) is accepted or rejected.
First-Order Logic (FO)

**Signature**: Predicate symbols \((P_1, \ldots, P_n)\) with arities \(k_1, \ldots, k_n\).

**Syntax** of FO:

\[
\varphi, \psi := P_i(x_1, \ldots, x_{k_i}) \mid \varphi \lor \psi \mid \varphi \land \psi \mid \neg \varphi \mid \exists x. \varphi \mid \forall x. \varphi
\]

**Semantics** of \(\varphi\):
Structure \((X, R_1, \ldots, R_n)\) is accepted or rejected.

**Example**: Directed graphs: one binary predicate \(E\).

<table>
<thead>
<tr>
<th>Graph class</th>
<th>Cliques</th>
<th>No node points to everyone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula</td>
<td>(\varphi = \forall x. \forall y. E(x, y))</td>
<td>(\psi = \neg \exists x. \forall y. E(x, y))</td>
</tr>
</tbody>
</table>

**Example graph**

Model of \(\varphi\)  
Model of \(\psi\)
Positive versus Monotone

**Goal**: Understand the role of negation in FO.
Positive versus Monotone

**Goal:** Understand the role of negation in FO.

**Positive formula:** no $\neg$

Monotone class of structures: closed under adding tuples to relations. For graph classes: monotone = closed under adding edges.

Example: graphs containing a triangle.

**Monotone formula** defines a monotone class of structures.

**Fact:** $\phi$ positive $\Rightarrow$ $\phi$ monotone.

What about the converse?

**Motivation:** Logics with fixed points. Fixed points need monotone $\phi$ $\Rightarrow$ positive $\phi$, syntactic condition.
Positive versus Monotone

**Goal**: Understand the role of negation in FO.

Positive formula: no ¬

Monotone class of structures: closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

Example: graphs containing a triangle.
Positive versus Monotone

**Goal**: Understand the role of negation in FO.

**Positive formula**: no \( \neg \)

**Monotone class of structures**: closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

**Example**: graphs containing a triangle.

**Monotone formula**: defines a monotone class of structures.
Positive versus Monotone

**Goal:** Understand the role of negation in FO.

**Positive formula:** no \(\neg\)

**Monotone class of structures:** closed under adding tuples to relations.

**For graph classes:** monotone = closed under adding edges.

**Example:** graphs containing a triangle.

**Monotone formula:** defines a monotone class of structures.

**Fact:** \(\varphi \) positive \(\Rightarrow\) \(\varphi \) monotone.
Positive versus Monotone

**Goal**: Understand the role of negation in FO.

**Positive formula**: no $\neg$

**Monotone class of structures**: closed under adding tuples to relations.

For graph classes: monotone $=$ closed under adding edges.

**Example**: graphs containing a triangle.

**Monotone formula**: defines a monotone class of structures.

**Fact**: $\varphi$ positive $\Rightarrow$ $\varphi$ monotone.

What about the converse?
Positive versus Monotone

**Goal:** Understand the role of negation in FO.

**Positive formula:** no \( \neg \)

**Monotone class of structures:** closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

**Example:** graphs containing a triangle.

**Monotone formula:** defines a monotone class of structures.

**Fact:** \( \varphi \) positive \( \Rightarrow \) \( \varphi \) monotone.

What about the converse?

**Motivation:** Logics with fixed points.
Fixed points need monotone \( \varphi \).
\( \rightarrow \) positive \( \varphi \), syntactic condition.
Lyndon’s theorem

Theorem (Lyndon 1959)

If $\varphi$ is monotone then $\varphi$ is equivalent to a positive formula.

On graph classes: FO-definable + monotone $\Rightarrow$ FO-definable without \neg.
Lyndon’s theorem

Theorem (Lyndon 1959)

If $\varphi$ is monotone then $\varphi$ is equivalent to a positive formula.

On graph classes: $\text{FO-definable} + \text{monotone} \Rightarrow \text{FO-definable without } \neg$.

⚠️ valid with infinite structures.
Lyndon’s theorem

Theorem (Lyndon 1959)

If \( \varphi \) is monotone then \( \varphi \) is equivalent to a positive formula.

On graph classes: FO-definable$\text{+}$monotone \( \Rightarrow \) FO-definable without \( \neg \).

\[\text{valid with } \text{infinite } \text{structures}.\]

What about finite structures?

This was open for 28 years...
Lyndon’s theorem

Theorem (Lyndon 1959)

If $\varphi$ is monotone then $\varphi$ is equivalent to a positive formula.

On graph classes: FO-definable+monotone $\Rightarrow$ FO-definable without $\neg$.

⚠️ valid with infinite structures.

What about finite structures?

This was open for 28 years...

**Theorem:** Lyndon’s theorem fails on finite structures:

- [Ajtai, Gurevich 1987]
  lattices, probas, number theory, complexity, topology, very hard

- [Stolboushkin 1995]
  EF games on grid-like structures, involved
Lyndon’s theorem

**Theorem (Lyndon 1959)**

If \( \varphi \) is monotone then \( \varphi \) is equivalent to a positive formula.

**On graph classes:** \( \text{FO-definable+monotone} \implies \text{FO-definable without } \neg. \)

⚠️ valid with infinite structures.

What about finite structures?

This was open for 28 years...

**Theorem:** Lyndon’s theorem fails on finite structures:

- [Ajtai, Gurevich 1987]
  lattices, probas, number theory, complexity, topology, very hard

- [Stolboushkin 1995]
  EF games on grid-like structures, involved

- [This work]
  EF games on words, elementary
Our results

**Finite Model Theory:**

Lyndon’s theorem fails on

- Finite words
- Finite graphs
- Finite structures (elementary proof), several versions:
  - one monotone predicate
  - some monotone predicates
  - all monotone predicates $=$ closure under surjective morphisms.

**Regular Language Theory:**

- Monotone FO languages $\neq$ Positive FO languages
- Algebraic characterization
- Logical characterization
- Decidable membership
- Undecidable membership
Our results

**Finite Model Theory:**

Lyndon’s theorem fails on

- **Finite words**
- **Finite graphs**
- **Finite structures** (elementary proof), several versions:
  - one monotone predicate
  - some monotone predicates
  - all monotone predicates $=$ closure under surjective morphisms.

**Regular Language Theory:**

<table>
<thead>
<tr>
<th>Monotone FO languages</th>
<th>$\neq$</th>
<th>Positive FO languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebraic characterization</td>
<td></td>
<td>Logical characterization</td>
</tr>
<tr>
<td>Decidable membership</td>
<td></td>
<td>Undecidable membership</td>
</tr>
</tbody>
</table>
FO on words, the usual way

Words on alphabet $A = \{ a, b[,\ldots] \}$: signature $(\leq, a, b[,\ldots])$

\[
\begin{array}{cccccc}
  a & b & a & a & b \\
  \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet
\end{array}
\]

- $x \leq y$: position $x$ before position $y$.
- $a(x)$: position $x$ labelled by letter $a$
FO on words, the usual way

Words on alphabet $A = \{a, b[, \ldots]\}$: signature ($\leq, a, b[, \ldots]$)

\[
\begin{array}{ccccc}
  a & b & a & a & b \\
  \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet
\end{array}
\]

- $x \leq y$ : position $x$ before position $y$.
- $a(x)$ : position $x$ labelled by letter $a$

Examples of formulas:
- $\exists x. a(x)$: Language $A^* a A^*$.
- $\exists x, y.(x \leq y \land a(x) \land b(y))$. Language $A^* a A^* b A^*$.
FO on words, the usual way

Words on alphabet $A = \{a, b, \ldots \}$: signature $(\leq, a, b, \ldots)$

$$\begin{array}{cccccc}
a & b & a & a & b \\
\bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet
\end{array}$$

- $x \leq y$: position $x$ before position $y$.
- $a(x)$: position $x$ labelled by letter $a$

Examples of formulas:
- $\exists x. a(x)$: Language $A^*aA^*$.
- $\exists x, y. (x \leq y \wedge a(x) \wedge b(y))$. Language $A^*aA^*bA^*$.

Theorem

First-order languages form a strict subclass of regular languages.

Example: $(aa)^*$ is not FO-definable. (Proof later)
Background: FO-definable languages

FO-definable languages are well-understood.

Theorem (Schützenberger, McNaughton, Papert)

A language \( L \subseteq A^* \) is FO-definable iff it is definable by:

- Star-free expression
- LTL
- Counter-free automaton

Intuition: FO languages are “Aperiodic”: cannot count modulo

\( L \) aperiodic: There is \( n \in \mathbb{N} \) such that \( \forall u, v, w \in A^*: uv^n w \in L \iff uv^{n+1} w \in L \).

Counter-free automaton:

Corollary: FO-definability is decidable for regular languages.
Background: FO-definable languages

FO-definable languages are well-understood.

**Theorem (Schützenberger, McNaughton, Papert)**

A language $L \subseteq A^*$ is FO-definable iff it is definable by:
- Star-free expression $\Leftrightarrow$ LTL $\Leftrightarrow$ counter-free automaton $\Leftrightarrow$ ...

Intuition: FO languages are “Aperiodic”: cannot count modulo $L$ aperiodic:
There is $n \in \mathbb{N}$ such that $\forall u, v, w \in A^*$:
$uv^n w \in L \Leftrightarrow uv^{n+1} w \in L$.

Corollary: FO-definability is decidable for regular languages.
Background: FO-definable languages

FO-definable languages are well-understood.

Theorem (Schützenberger, McNaughton, Papert)

A language \( L \subseteq A^* \) is FO-definable iff it is definable by:
- Star-free expression \( \iff \) LTL \( \iff \) counter-free automaton \( \iff \ldots \)

**Intuition:** FO languages are “Aperiodic”: cannot count modulo

\( L \) _aperiodic:_ There is \( n \in \mathbb{N} \) such that \( \forall u, v, w \in A^*: \)

\[
uv^n w \in L \iff uv^{n+1} w \in L.
\]
Background: FO-definable languages

FO-definable languages are well-understood.

**Theorem (Schützenberger, McNaughton, Papert)**

A language $L \subseteq A^*$ is FO-definable iff it is definable by:
- Star-free expression $\Leftrightarrow$ LTL $\Leftrightarrow$ counter-free automaton $\Leftrightarrow$ ...

**Intuition**: FO languages are “Aperiodic”: cannot count modulo $L$ aperiodic: There is $n \in \mathbb{N}$ such that $\forall u, v, w \in A^*$:

$$uv^n w \in L \Leftrightarrow uv^{n+1} w \in L.$$  

$\Leftrightarrow$ Counter-free automaton: No cycle of the form:
Background: FO-definable languages

FO-definable languages are well-understood.

Theorem (Schützenberger, McNaughton, Papert)

A language \( L \subseteq A^* \) is FO-definable iff it is definable by:

Star-free expression \( \iff \) LTL \( \iff \) counter-free automaton \( \iff \) ...

Intuition: FO languages are “Aperiodic”: cannot count modulo

\( L \) aperiodic: There is \( n \in \mathbb{N} \) such that \( \forall u, v, w \in A^* \):

\[
uv^n w \in L \iff uv^{n+1} w \in L.
\]

\( \iff \) Counter-free automaton: No cycle of the form:

\[
\begin{array}{c}
\text{Corollary: } \text{FO-definability is decidable for regular languages.}
\end{array}
\]
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).
For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).

\(a, b, \ldots\) now independent.
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).

\(a, b, \ldots\) now independent.

\(\rightarrow\) Words on alphabet \(\mathcal{P}\{a, b, \ldots\}\):

\[
\emptyset \quad \{b\} \quad \{a, b\} \quad \emptyset \quad \{b\}
\]

\[
\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet
\]

We will note \(\Sigma = \{a, b, \ldots\}\), and \(A = \mathcal{P}(\Sigma)\) the alphabet.

- Useful e.g. in verification (LTL,\ldots):
  independent signals can be true or false simultaneously.
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).

\(a, b, \ldots\) now independent.

\(\rightarrow\) Words on alphabet \(\mathcal{P}(\{a, b, \ldots\})\):

\[
\emptyset \quad \{b\} \quad \{a, b\} \quad \emptyset \quad \{b\}
\]

We will note \(\Sigma = \{a, b, \ldots\}\), and \(A = \mathcal{P}(\Sigma)\) the alphabet.

- Useful e.g. in verification (LTL,\ldots):
  independent signals can be true or false simultaneously.

- FO languages on alphabet \(A\) are the same (\(\text{Preds} = \Sigma\) or \(A\)).
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).

\(a, b, \ldots\) now independent.

\(\rightarrow\) Words on alphabet \(\mathcal{P}(\{a, b, \ldots\})\):

\[
\begin{array}{cccccc}
\emptyset & \{b\} & \{a, b\} & \emptyset & \{b\} \\
\bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet
\end{array}
\]

We will note \(\Sigma = \{a, b, \ldots\}\), and \(A = \mathcal{P}(\Sigma)\) the alphabet.

- Useful e.g. in verification (LTL,\ldots):
  independent signals can be true or false simultaneously.

- FO languages on alphabet \(A\) are the same (Preds=\(\Sigma\) or \(A\)).

- We no longer have \(\neg a(x) \equiv \bigvee_{\beta \neq a} \beta(x)\).
FO on words, the “unconstrained” way

For now, a word is a structure \( (X, \leq, a, b, \ldots) \) where

- \( \leq \) is a total order
- \( a, b, \ldots \) form a partition of \( X \).

\( a, b, \ldots \) now independent.

→ Words on alphabet \( \mathcal{P}(\{a, b, \ldots\}) \):

\[
\emptyset \quad \{b\} \quad \{a, b\} \quad \emptyset \quad \{b\}
\]

\[
\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet
\]

We will note \( \Sigma = \{a, b, \ldots\} \), and \( A = \mathcal{P}(\Sigma) \) the alphabet.

- Useful e.g. in verification (LTL,\ldots):
  independent signals can be true or false simultaneously.

- FO languages on alphabet \( A \) are the same (Preds=\( \Sigma \) or \( A \)).

- We no longer have \( \neg a(x) \equiv \bigvee_{\beta \neq a}^{} \beta(x) \).
  → Negation necessary for full FO.
The $\text{FO}^+$ logic: positive formulas

$\text{FO}^+$ Logic: $a$ ranges over $\Sigma$, no $\neg$

$\varphi, \psi ::= a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi$
The $\mathbf{FO}^+$ logic: positive formulas

$\mathbf{FO}^+$ Logic: $a$ ranges over $\Sigma$, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi$$

Example: On $\Sigma = \{a, b\}$:

$$\exists x, y. (x \leq y) \land a(x) \land b(y) \rightsquigarrow (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$$
The $\text{FO}^+$ logic: positive formulas

$\text{FO}^+$ Logic: $a$ ranges over $\Sigma$, no $\neg$

\[ \varphi, \psi := a(x) | x \leq y | x < y | \varphi \lor \psi | \varphi \land \psi | \exists x. \varphi | \forall x. \varphi \]

**Example:** On $\Sigma = \{a, b\}$:

\[ \exists x, y. (x \leq y) \land a(x) \land b(y) \leadsto (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*) \]

**Remark:** $\emptyset^*$ undefinable in $\text{FO}^+$ (cannot say "$\neg a"$).
The **FO**\(^+\) logic: positive formulas

**FO**\(^+\) Logic: \( a \) ranges over \( \Sigma \), no \( \neg \)

\[ \varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi \]

**Example:** On \( \Sigma = \{a, b\} \):

\[ \exists x, y. (x \leq y) \land a(x) \land b(y) \rightsquigarrow (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*) \]

**Remark:** \( \emptyset^* \) undefinable in **FO**\(^+\) (cannot say "\( \neg a \)").

More generally: **FO**\(^+\) can only define monotone languages:

\[ u\alpha v \in L \text{ and } \alpha \subseteq \beta \Rightarrow u\beta v \in L \]
The **FO**\(^{+}\) **logic: positive formulas**

**FO**\(^{+}\) **Logic:** \(a\) ranges over \(\Sigma\), no \(\neg\)

\[
\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi
\]

**Example:** On \(\Sigma = \{a, b\}\):

\[
\exists x, y. (x \leq y) \land a(x) \land b(y) \leadsto (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)
\]

**Remark:** \(\emptyset^*\) undefinable in \(\text{FO}^{+}\) (cannot say "\(\neg a\)").

More generally: \(\text{FO}^{+}\) can only define **monotone languages**:

\[
u\alpha v \in L \text{ and } \alpha \subseteq \beta \Rightarrow u\beta v \in L
\]

**Motivation:** abstraction of many logics not closed under \(\neg\).
The $\text{FO}^+$ logic: positive formulas

$\text{FO}^+$ Logic: $a$ ranges over $\Sigma$, no $\neg$

$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x.\varphi \mid \forall x.\varphi$

Example: On $\Sigma = \{a, b\}$:

$\exists x, y. (x \leq y) \land a(x) \land b(y) \rightsquigarrow (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$

Remark: $\emptyset^*$ undefinable in $\text{FO}^+$ (cannot say "$\neg a$").

More generally: $\text{FO}^+$ can only define monotone languages:

$u\alpha v \in L$ and $\alpha \subseteq \beta \Rightarrow u\beta v \in L$

Motivation: abstraction of many logics not closed under $\neg$.

Question [Colcombet]: FO & monotone $\Rightarrow$ $\text{FO}^+$
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not $\text{FO}^+$-definable.
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not $\text{FO}^+$-definable.

Alphabet $A = \{\emptyset, a, b, c, (a)_b, (b)_c, (c)_a, (a/b)_c\}$. Let $a^\uparrow = \{a, (a)_b, (c)_a\}$. 
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not $\text{FO}^+$-definable.

Alphabet $A = \{\emptyset, a, b, c, (a), (b), (c), (a^b)^c\}$. Let $a^\uparrow = \{a, (a)^b, (c)^a\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup A^* (a^b)^c A^*$. 
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not $\text{FO}^+$-definable.

Alphabet $A = \{\emptyset, a, b, c, (a), (b), (c), (a, b, c), (b, c, a), (a, b, c, a)\}$. Let $a^\uparrow = \{a, (b), (c)\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \bigcup A^* (a \frac{b}{c}) A^*$. Monotone
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not $\text{FO}^+$-definable.

Alphabet $A = \{\emptyset, a, b, c, (a)_b, (b)_c, (c)_a, (a\ b\ c), (b\ c), (c\ a), (a\ b\ c)\}$. Let $a^\uparrow = \{a, (a)_b, (c)_a\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup A^* \left(\left(\begin{array}{c} a \\ b \\ c \end{array}\right)\right) A^*$. Monotone

Lemma: $L$ is FO-definable.

Proof: is counter-free. (no cycle labelled $\nu \geq 2$)
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not FO$^+$-definable.

Alphabet $A = \{\emptyset, a, b, c, (a)_b, (b)_c, (c)_a, (a\,\,b\,\,c)_c\}$. Let $a^\uparrow = \{a, (a)_b, (c)_a\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup A^* (a\,\,b\,\,c)_c A^*$. Monotone

Lemma: $L$ is FO-definable.

Proof: is counter-free. (no cycle labelled $\nu \geq 2$)

To prove $L$ is not FO$^+$-definable: Ehrenfeucht-Fraïssé games.
Ehrenfeucht-Fraïssé games for FO

**Definition (EF games)**

Played on two words $u$, $v$. At each round $i$:

- **Spoiler** places token $i$ in $u$ or $v$.
- **Duplicator** must answer token $i$ in the other word such that
  - same letter for token $i$,
  - same order between tokens.

We note $u \equiv_n v$ if Duplicator can survive $n$ rounds on $u$, $v$. 

**Theorem (Ehrenfeucht, Fraïssé, 1950-1961)**

$L$ not FO-definable $\iff$ For all $n$, there are $u \in L$, $v \not\in L$ s.t. $u \equiv_n v$.

**Example**

Proving $(aa)^*$ is not FO-definable:

- $u = a_2k \in (aa)^*$: $a \ a \ a \ a \ a \ a \ a \ a \ a \ a$
- $v = a_2k-1 \not\in (aa)^*$: $a \ a \ a \ a \ a \ a \ a \ a \ a \ a \ 1$
Ehrenfeucht-Fraïssé games for FO

Definition (EF games)

Played on two words $u, v$. At each round $i$:

- **Spoiler** places token $i$ in $u$ or $v$.
- **Duplicator** must answer token $i$ in the other word such that
  - same letter for token $i$,
  - same order between tokens.

We note $u \equiv_n v$ if **Duplicator** can survive $n$ rounds on $u, v$. 

Theorem (Ehrenfeucht, Fraïssé, 1950-1961)

$L$ not FO-definable $\iff$ For all $n$, there are $u \in L$, $v \not\in L$ s.t. $u \equiv_n v$. 

Example

Proving $(aa)^*$ is not FO-definable:

$u = a^k \in (aa)^*$: $a a a a a a a a a a$

$v = a^{k-1} \not\in (aa)^*$: $a a a a a a a a a$
Ehrenfeucht-Fraïssé games for FO

**Definition (EF games)**
Played on two words $u, v$. At each round $i$:
- **Spoiler** places token $i$ in $u$ or $v$.
- **Duplicator** must answer token $i$ in the other word such that
  - same letter for token $i$,
  - same order between tokens.

We note $u ≡_n v$ if Duplicator can survive $n$ rounds on $u$, $v$.

**Theorem (Ehrenfeucht,Fraïssé, 1950-1961)**
$L$ not FO-definable $⇔$ For all $n$, there are $u ∈ L$, $v ∉ L$ s.t. $u ≡_n v$. 
Ehrenfeucht-Fraïssé games for FO

Definition (EF games)
Played on two words $u, v$. At each round $i$:

- **Spoiler** places token $i$ in $u$ or $v$.
- **Duplicator** must answer token $i$ in the other word such that
  - same letter for token $i$,
  - same order between tokens.

We note $u \equiv_n v$ if Duplicator can survive $n$ rounds on $u, v$.

**Theorem (Ehrenfeucht,Fraïssé, 1950-1961)**
$L$ not FO-definable $\iff$ For all $n$, there are $u \in L$, $v \notin L$ s.t. $u \equiv_n v$.

**Example**
Proving $(aa)^*$ is not FO-definable:

$$u = a^{2k} \quad \in (aa)^*: \quad a \ a \ a \ a \ a \ a \ a \ a \ a$$

$$v = a^{2k-1} \quad \notin (aa)^*: \quad a \ a \ a \ a \ a \ a \ a \ a \ a$$
Proving $\mathbf{FO}^+$-undefinability

Definition ($\mathbf{EF}^+$ games)
Previous rule: $a \text{ in } u \iff a \text{ in } v$.

We write $u \leq_n v$ if Duplicator can survive $n$ rounds.

Theorem (Correctness of $\mathbf{EF}^+$ games)

$L$ not $\mathbf{FO}^+-$definable $\iff \forall n, \exists u \in L, v \not\in L$ s.t. $u \leq_n v$.

[Stolboushkin 1995+this work]

Application: Proving $L$ is not $\mathbf{FO}^+-$definable

$u \in L$:

```
a b c a b c a b c
```

$v \not\in L$:

```
(a b)(b c)(c a)(a b)(b c)(c a)(a b)(b c)
```
Proving $\text{FO}^+$-undefinability

Definition ($\text{EF}^+$ games)

New rule: $a \text{ in } u \Rightarrow a \text{ in } v$.

We write $u \preceq_n v$ if Duplicator can survive $n$ rounds.
Proving $\text{FO}^+$-undefinability

**Definition ($\text{EF}^+$ games)**

New rule: $a \text{ in } u \Rightarrow a \text{ in } v$.

We write $u \preceq_n v$ if Duplicator can survive $n$ rounds.

**Theorem (Correctness of $\text{EF}^+$ games)**

$L$ not $\text{FO}^+$-definable $\iff \forall n$, there are $u \in L$, $v \notin L$ s.t. $u \preceq_n v$.

[Stolboushkin 1995 + this work]
Proving $\text{FO}^+\text{-undefinability}$

**Definition (EF$^+$ games)**

New rule: $a \in u \Rightarrow a \in v$.

We write $u \leq_n v$ if Duplicator can survive $n$ rounds.

**Theorem (Correctness of EF$^+$ games)**

$L$ not $\text{FO}^+$-definable $\iff \forall n$, there are $u \in L$, $v \notin L$ s.t. $u \leq_n v$.

[Stolboushkin 1995 + this work]

**Application: Proving $L$ is not $\text{FO}^+$-definable**

$u \in L : \quad a \quad b \quad c \quad a \quad b \quad c \quad a \quad b \quad c$

$v \notin L : \quad (a) \quad (b) \quad (c) \quad (a) \quad (b) \quad (c) \quad (a) \quad (b) \quad (c)$
From finite words to finite structures.

Goal: Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**
Axiomatize in FO that $\leq$ is a total order.
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**

Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are monotone but not $\leq$. 
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**
Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**
Alphabet encoded by one binary predicate $A$.

\[
a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)
\]
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature ($\leq, a, b, c$) assuming $\leq$ is a total order.

**Several monotone predicates**

Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**

Alphabet encoded by one binary predicate $A$.

$$
a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)
$$

$A$ is monotone but not $\leq$. 

---

All monotone predicates = closure under surjective morphisms

Problem: We cannot say that $\leq$ is total in a monotone way.

Solution: Introduce a predicate $\not\leq$.

▶ Require $\forall x, y. (x \leq y) \lor (x \not\leq y)$

▶ If $\exists x, y. (x \leq y) \land (x \not\leq y) \rightarrow \text{accept}$

▶ Axiomatize that $\leq$ is total assuming $\not\leq$ is its complement.
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.  
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**

Axiomatize in FO that $\leq$ is a total order.  
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**

Alphabet encoded by one binary predicate $A$.  
\[
a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)
\]

$A$ is monotone but not $\leq$.

**All monotone predicates = closure under surjective morphisms**

**Problem:** We cannot say that $\leq$ is total in a monotone way.
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**
Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**
Alphabet encoded by one binary predicate $A$.
\[
a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)
\]
$A$ is monotone but not $\leq$.

**All monotone predicates = closure under surjective morphisms**

**Problem:** We cannot say that $\leq$ is total in a monotone way.
**Solution:** Introduce a predicate $\not\leq$. 

\[\forall x, y. (x \leq y) \lor (x \not\leq y)\]
\[\forall x, y. (x \leq y) \land (x \not\leq y) \rightarrow \text{accept}\]
\[\text{Axiomatize that } \leq \text{ is total assuming } \not\leq \text{ is its complement.}\]
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.  
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**  
Axiomatize in FO that $\leq$ is a total order.  
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**  
Alphabet encoded by one binary predicate $A$.  
\[
a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)
\]  
$A$ is monotone but not $\leq$.

**All monotone predicates = closure under surjective morphisms**  
**Problem:** We cannot say that $\leq$ is total in a monotone way.  
**Solution:** Introduce a predicate $\not\leq$.

- Require $\forall x, y. (x \leq y) \lor (x \not\leq y)$
- If $\exists x, y. (x \leq y) \land (x \not\leq y)$ → accept
- Axiomatize that $\leq$ is total assuming $\not\leq$ is its complement.
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**
Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**
Alphabet encoded by one binary predicate $A$.

$$a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)$$

$A$ is monotone but not $\leq$.

**All monotone predicates $=$ closure under surjective morphisms**
Problem: We cannot say that $\leq$ is total in a monotone way.
Solution: Introduce a predicate $\nleq$.

- Require $\forall x, y. (x \leq y) \lor (x \nleq y)$
- If $\exists x, y. (x \leq y) \land (x \nleq y) \rightarrow$ accept
- Axiomatize that $\leq$ is total assuming $\nleq$ is its complement.

$a, b, c, \leq, \nleq$ are monotone.
From finite words to finite graphs
From finite words to finite graphs

Encode words into (directed) graphs, here $ab^a_b c$:
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(a) c$:

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a^* b^* c^*) \cup (A^* (\begin{array}{c} a \\ b \\ c \end{array}) A^*)$.

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a^* b^* c^*) \cup (A^* (\begin{array}{c} a \\ b \\ c \end{array}) A^*)$. 

Left as exercise: Same with undirected graphs.
From finite words to finite graphs

Encode words into (directed) graphs, here $ab^{a,b,c}$:

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup (A^* \begin{pmatrix}a \\ b \\ c \end{pmatrix} A^*)$.

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(b^a)c$:

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup (A^* \left(\begin{array}{c} a \\ b \\ c \end{array}\right) A^*)$.

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(b^a)c$:

$\xrightarrow{\quad}$ formula $\psi_L$ for graphs encoding words of $L = (a^+ b^+ c^+)^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $\xrightarrow{\quad}$
  - $\xrightarrow{\quad}$
  - $\xrightarrow{\quad}$

Left as exercise: Same with undirected graphs.
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(b^a)c$:

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a^{\uparrow}b^{\uparrow}c^{\uparrow})^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $x_a \rightarrow x_b \rightarrow x_c$
  - $\square \rightarrow \square$
  - $\square \rightarrow \square$

- $\psi^+$ is a disjunction of excess edges:
From finite words to finite graphs

Encode words into (directed) graphs, here $ab^a_b c$:

\[
\begin{align*}
\psi_L & \text{ formula for graphs encoding words of } L = (a^* b^* c^*) \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*). \\
\text{Rule out other graphs, in a monotone way:} & \\
\text{\quad } \psi^- & \text{ is a conjunction of edge requirements:} \\
\text{\quad \quad } & x_a \quad x_b \quad x_c \\
\text{\quad \quad } & \square \quad \square, \ldots \\
\text{\quad } \psi^+ & \text{ is a disjunction of excess edges:} \\
\text{\quad \quad } & x_a \quad x_b \\
\end{align*}
\]
From finite words to finite graphs

Encode words into (directed) graphs, here $ab^a_c$:

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a^+ b^+ c^+) \cup (A^+ \left( \begin{array}{c} a \\ b \\ c \end{array} \right) A^+)$. Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $\rightarrow$ $x_a \rightarrow x_b \rightarrow x_c$,
  - $\rightarrow$ ,
  - $\rightarrow$ ,

- $\psi^+$ is a disjunction of excess edges:
  - $\rightarrow$ $x_a \rightarrow x_b$,
  - $\rightarrow$ ,
  - $\rightarrow$ ,
From finite words to finite graphs

Encode words into (directed) graphs, here $ab^c$:

$$x_a \rightarrow x_b \rightarrow x_c$$

→ formula $\psi_L$ for graphs encoding words of $L = (a^* b^* c^*) \cup (A^* (\begin{array}{c} a \\ b \\ c \end{array}) A^*)$.

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $x_a \rightarrow x_b \rightarrow x_c$,
  - $\square \rightarrow \square, \ldots$

- $\psi^+$ is a disjunction of excess edges:
  - $x_a \rightarrow x_b$,
  - $\square \rightarrow \square, \ldots$

**Final Formula:** $\exists x_a, x_b, x_c. (\psi^- \land (\psi_L \lor \psi^+))$
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(b^a)c$:

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a^{b^a}b^{b^c})* \cup (A^* (b^a_c) A^*)$. Rule out other graphs, in a monotone way:

$\rightarrow$ $\psi^-$ is a conjunction of edge requirements:

$\rightarrow$ $\psi^+$ is a disjunction of excess edges:

Final Formula: $\exists x_a, x_b, x_c. (\psi^- \land (\psi_L \lor \psi^+))$

Left as exercise: Same with undirected graphs.
Back to regular languages

Theorem

Given $L$ regular on an ordered alphabet, it is decidable whether

- $L$ is monotone (e.g. automata inclusion)
- $L$ is FO-definable [Schützenberger, McNaughton, Papert]

Can we decide whether $L$ is $\text{FO}^+$-definable?
Back to regular languages

Theorem
Given $L$ regular on an ordered alphabet, it is **decidable** whether

- $L$ is monotone (e.g. automata inclusion)
- $L$ is FO-definable [Schützenberger, McNaughton, Papert]

Can we decide whether $L$ is FO$^+$-definable?

**Theorem**

FO$^+$-definability is **undecidable** for regular languages.
Back to regular languages

Theorem
Given $L$ regular on an ordered alphabet, it is decidable whether

- $L$ is monotone (e.g. automata inclusion)
- $L$ is FO-definable [Schützenberger, McNaughton, Papert]

Can we decide whether $L$ is FO$^+$-definable?

Theorem

FO$^+$-definability is undecidable for regular languages.

Reduction from Turing Machine Mortality:
A deterministic TM $M$ is mortal if there a uniform bound $n$ on the runs of $M$ from any configuration.

Undecidable [Hooper 1966].
#### Undecidability proof sketch

Given a TM $M$, we build a regular language $L$ such that

$$M \text{ mortal } \iff L \text{ is } \text{FO}^+\text{-definable.}$$
Undecidability proof sketch

Given a TM $M$, we build a regular language $L$ such that

$$M \text{ mortal } \iff L \text{ is } \text{FO}^+\text{-definable.}$$

Building $L$:
Inspired from $(a^{\uparrow}b^{\uparrow}c^{\uparrow})^*$, but:

- $a, b, c \rightsquigarrow$ Words from languages $C_1, C_2, C_3$ encoding configs of $M$.
- All transitions of $M$ follow the cycle:

$$(a) \rightsquigarrow (a_{u_1}), (b), (c) \rightsquigarrow (u_{u_2}), \text{ exists iff } u_1 \xrightarrow{M} u_2.$$
Undecidability proof sketch

Given a TM $M$, we build a regular language $L$ such that

$$M \text{ mortal } \iff L \text{ is } \text{FO}^+\text{-definable.}$$

Building $L$:
Inspired from $(a^{↑}b^{↑}c^{↑})^*$, but:

- $a, b, c \leadsto$ Words from languages $C_1, C_2, C_3$ encoding configs of $M$.

- All transitions of $M$ follow the cycle:

$$\begin{array}{c}
C_1 \\
\leftarrow \\
\rightarrow \\
C_3
\end{array}$$

- $(a, b, c) \leadsto (u_1, u_2)$, exists iff $u_1 \xrightarrow{M} u_2$.

We choose

$$L := (C_1^{↑} \cdot C_2^{↑} \cdot C_3^{↑})^*$$
Undecidability proof sketch

Given a TM $M$, we build a regular language $L$ such that

$$M \text{ mortal } \iff L \text{ is } \text{FO}^+\text{-definable.}$$

**Building $L$:**

Inspired from $(a^\uparrow b^\uparrow c^\uparrow)^*$, but:

- $a, b, c \sim$ Words from languages $C_1, C_2, C_3$ encoding configs of $M$.

- All transitions of $M$ follow the cycle:

```
C_1 ←→ C_2 ←→ C_3
```

- $(a^\uparrow), (b^\uparrow), (c^\uparrow) \sim (u_1^\uparrow, u_2^\uparrow)$, exists iff $u_1 \xrightarrow{M} u_2$.

We choose

$$L := (C_1^\uparrow \cdot C_2^\uparrow \cdot C_3^\uparrow)^*$$

$u \in L \nRightarrow u$ encodes a run of $M$. 
The reduction

If $M$ not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of $M$, and play Duplicator in :

\[
\begin{align*}
u \in L :& \quad u_1 \ u_2 \ u_3 \ \ldots \ \ u_{n-1} \ u_n \\
v \not\in L :& \quad (u_1) \ (u_2) \ (u_3) \ \ldots \ (u_{n-1}) \\
\end{align*}
\]

$\rightarrow L$ is not $\text{FO}^+\text{-definable.}$
The reduction

**If $M$ not mortal:**
Let $u_1, u_2, \ldots, u_n$ a long run of $M$, and play **Duplicator** in:

\[
\begin{align*}
&\text{ } u \in L : \quad u_1 \quad u_2 \quad u_3 \quad \ldots \quad u_{n-1} \quad u_n \\
&\text{ } v \notin L : \quad (u_1) \quad (u_2) \quad (u_3) \quad \ldots \quad (u_{n-1})
\end{align*}
\]

$\rightarrow L$ is not $\text{FO}^+$-definable.

**If $M$ mortal with bound $n$:**
Abstract $u_i$ by the length of the run of $M$ starting in it (at most $n$).
The reduction

If $M$ not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of $M$, and play Duplicator in:

$$u \in L: \quad u_1 \quad u_2 \quad u_3 \quad \ldots \quad u_{n-1} \quad u_n$$

$$v \notin L: \quad \left(\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{array}\right)$$

$\rightarrow$ $L$ is not $\text{FO}^+$-definable.

If $M$ mortal with bound $n$:
Abstract $u_i$ by the length of the run of $M$ starting in it (at most $n$).

Play Spoiler in the abstracted game (here $n = 5$):

\[
\begin{array}{ccccccccccc}
u : & 2 & 3 & 2 & 4 & 3 & 5 & 4 & 3 & 4 & 4 \\
u : & \left(\begin{array}{c} 2 \\ 1 \end{array}\right) & \left(\begin{array}{c} 3 \\ 2 \end{array}\right) & \left(\begin{array}{c} 2 \\ 1 \end{array}\right) & \left(\begin{array}{c} 4 \\ 3 \end{array}\right) & \left(\begin{array}{c} 3 \\ 2 \end{array}\right) & \left(\begin{array}{c} 5 \\ 4 \end{array}\right) & \left(\begin{array}{c} 4 \\ 3 \end{array}\right) & \left(\begin{array}{c} 5 \\ 4 \end{array}\right) & \left(\begin{array}{c} 5 \\ 4 \end{array}\right) & \left(\begin{array}{c} 5 \\ 4 \end{array}\right)
\end{array}
\]

$\rightarrow$ $L$ is $\text{FO}^+$-definable.
The reduction

If $M$ not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of $M$, and play Duplicator in:

$$u \in L : \quad u_1 \quad u_2 \quad u_3 \quad \ldots \quad u_{n-1} \quad u_n$$

$$v \notin L : \quad (u_1) \quad (u_2) \quad (u_3) \quad \ldots \quad (u_{n-1})$$

$\rightarrow L$ is not $\text{FO}^+$-definable.

If $M$ mortal with bound $n$:
Abstract $u_i$ by the length of the run of $M$ starting in it (at most $n$).

Play Spoiler in the abstracted game (here $n = 5$):

$$u : \quad 2 \quad 3 \quad 2 \quad 4 \quad 3 \quad 5 \quad 4 \quad 3 \quad 4 \quad 4$$

$$v : \quad (2) \quad (3) \quad (2) \quad (4) \quad (3) \quad (5) \quad (4) \quad (5)$$

Spoiler always wins in $2n$ rounds $\rightarrow L$ is $\text{FO}^+$-definable.
Ongoing work

With Thomas Colcombet:
Exploring the consequences of this in other frameworks:
  ▶ regular cost functions,
  ▶ logics on linear orders,
  ▶ ...

With Quentin Moreau:
  ▶ Links with LTL
  ▶ FO2 fragment
  ▶ ...

Slogan:
FO variants without negation will often display this behaviour.
Ongoing work

With Thomas Colcombet:
Exploring the consequences of this in other frameworks:
- regular cost functions,
- logics on linear orders,
- ...

With Quentin Moreau:
- Links with LTL
- FO2 fragment
- ...

Slogan:
FO variants without negation will often display this behaviour.

Thanks for your attention!